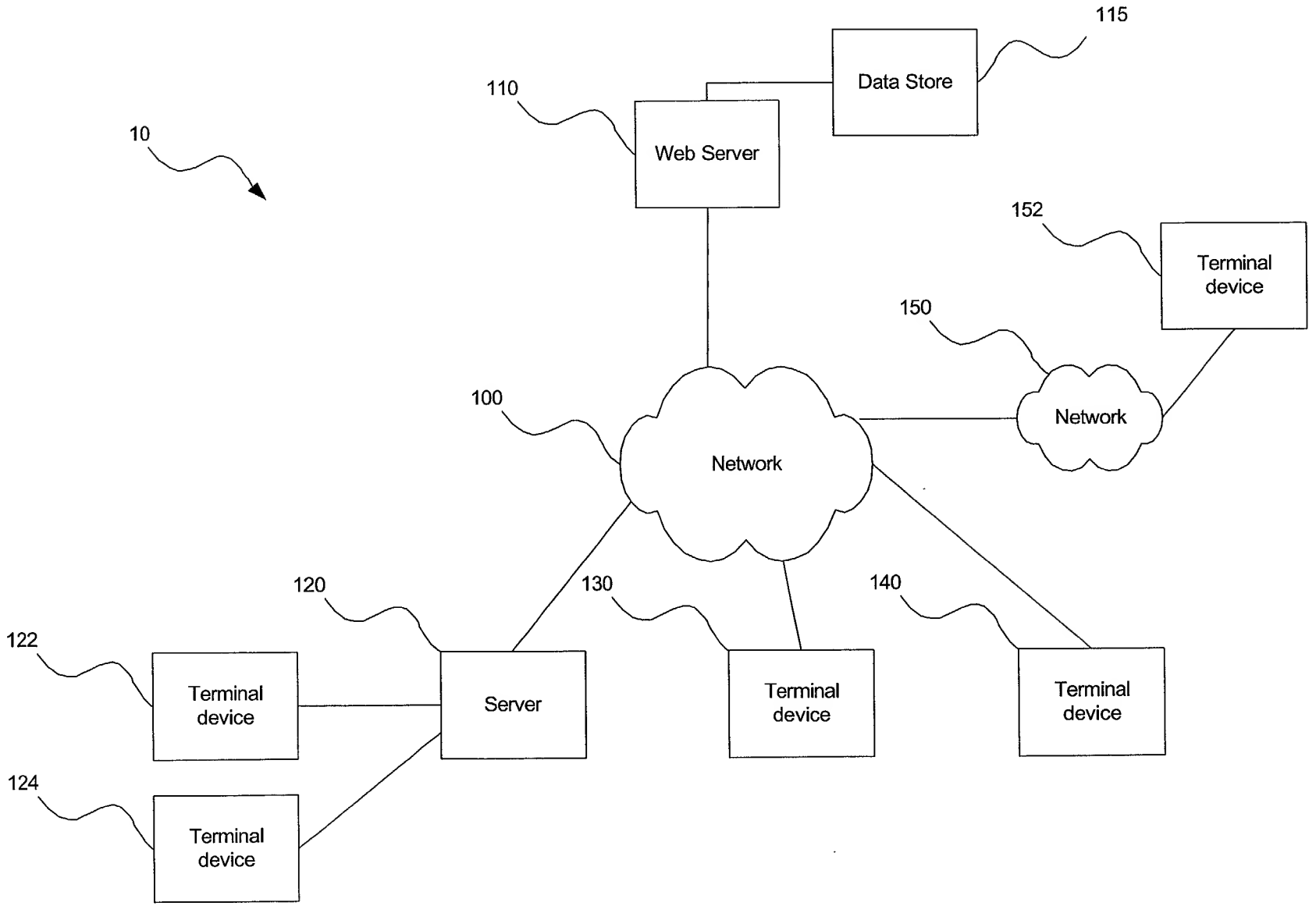
10
115
Data Store
110
Web Server
152
Terminal device
150
Network
100
Network
120
130
140
122
Terminal device
Server
Terminal device
Terminal device
124
Terminal device

FIG. 1

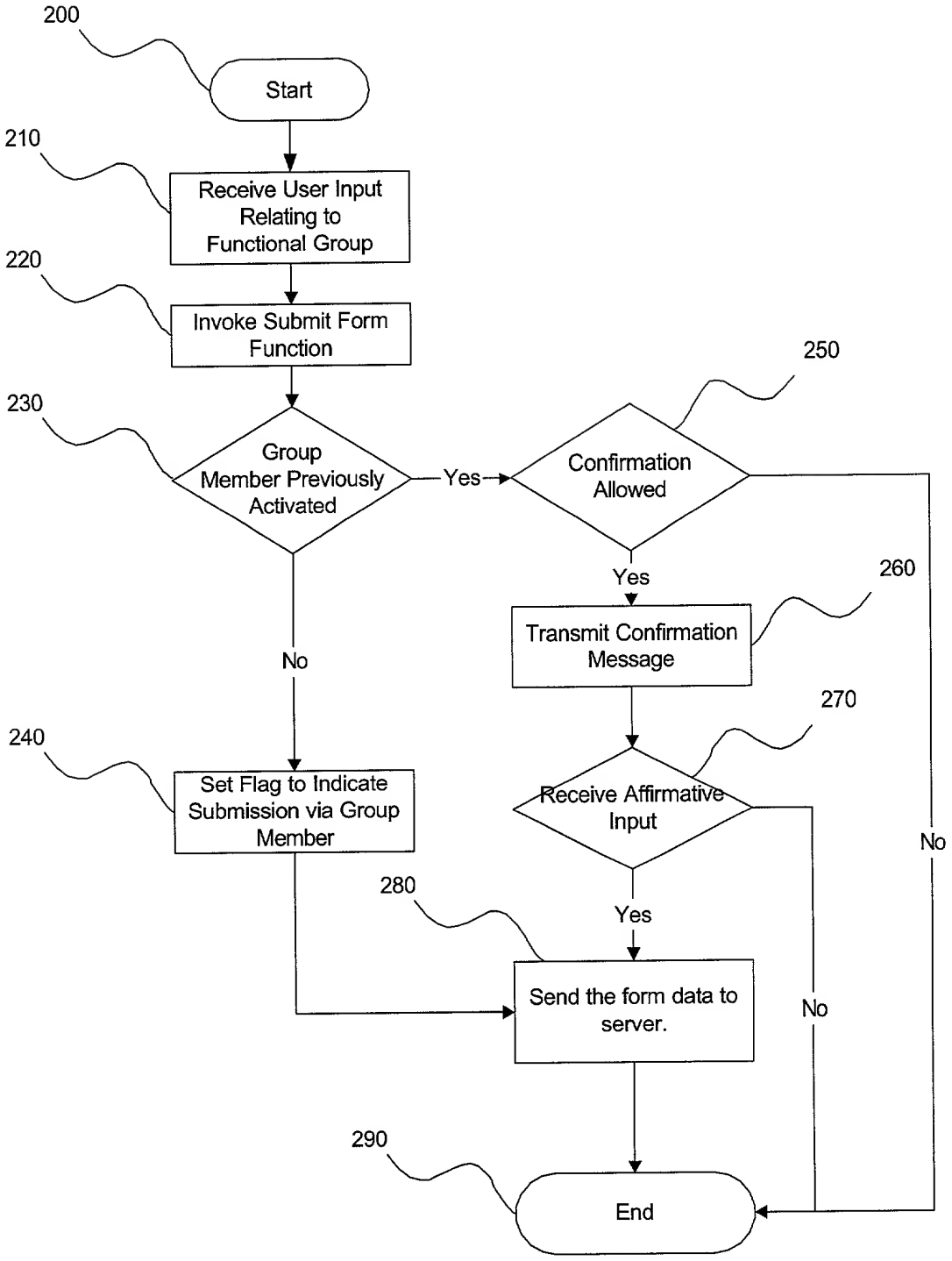
200
Start
210
Receive User Input Relating to Functional Group
220
Invoke Submit Form Function
230
Group Member Previously Activated
Yes
250
Confirmation Allowed
Yes
260
Transmit Confirmation Message
270
Receive Affirmative Input
Yes
280
Send the form data to server.
No
No
240
Set Flag to Indicate Submission via Group Member
No
290
End

FIG. 2

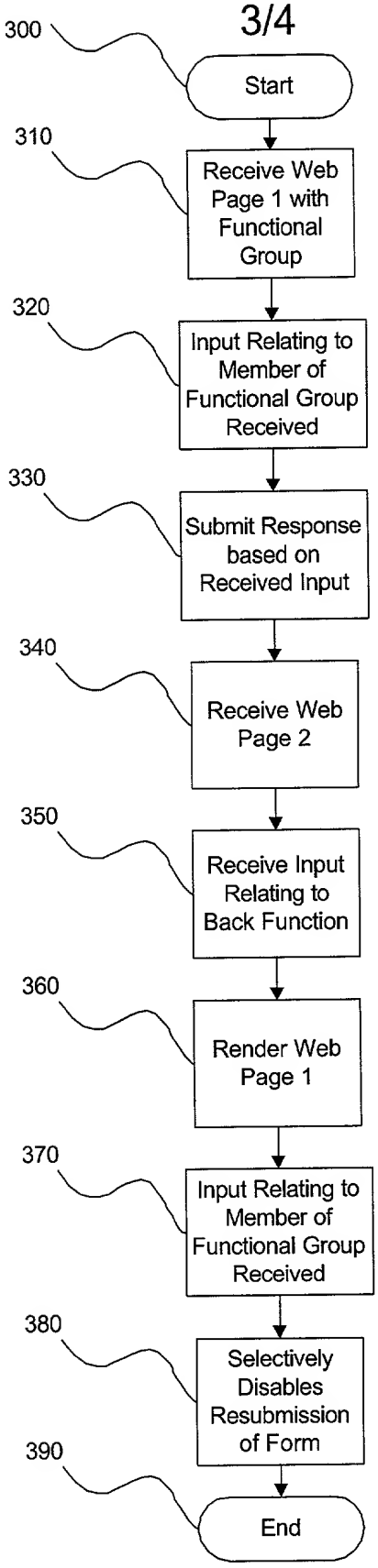
300
Start
310
Receive Web Page 1 with Functional Group
320
Input Relating to Member of Functional Group Received
330
Submit Response based on Received Input
340
Receive Web Page 2
350
Receive Input Relating to Back Function
360
Render Web Page 1
370
Input Relating to Member of Functional Group Received
380
Selectively Disables Resubmission of Form
390
End

FIG. 3

```
/*============================================================================
        PageHistory methods

        Description:
            definition of the PageHistory methods

============================================================================ */
function PageHistory_setPageFlag(name, flag) {
        //variable used in the cookie: literal string with ' to escape the potential : inside of name
        var namelit = "'" + name + "'";
        //set it in the cookie string
        //case in it is in pages already, then modify value in cookie
        if (!this.isNotInHistory(name)) {
                var repl = new RegExp(namelit + ":" + "[^\\s]*\\s");
                debug(GenDebug, "this.cookie before", this.cookie);
                debug(GenDebug, "repl", repl);
                this.cookie = this.cookie.replace(repl, namelit + ":" + flag + " ");
                debug(GenDebug, "this.cookie after", this.cookie);
        }
        else {
                //case cookie empty, create it
                if (this.cookie == "") {
                        this.cookie = "{ " + namelit + ":" + flag + " }";
                }
                //else add it to the beginning
                else {
                        this.cookie = "{ " + namelit + ":" + flag + " , " + this.cookie.substr(1);
                }
                this.size++;
                debug(GenDebug, "this.size", this.size);
                //case size > maxsize, remove lru = end
                if (this.size > this.maxsize) {
                        debug(GenDebug, "this.cookie before size restraint", this.cookie);
                        this.cookie = this.cookie.slice(0, this.cookie.lastIndexOf(",")) + "}";
                        debug(GenDebug, "this.cookie after size restraint", this.cookie);
                }
        }
        debug(GenDebug, "this.cookie end", this.cookie);
        //set it in the hashtable
        this.pages[name] = flag;
        //set the cookie
        setCookie("pageh", this.cookie);
}

function PageHistory_getPageFlag(name) {
        if (IsDef(this.pages[name])) return this.pages[name];
}
function PageHistory_isNotInHistory(name) {
        if (IsUndef(this.pages[name])) return true;
        return false;
}

/*============================================================================
        PageHistory()

        Description:
```

FIG. 4

```
        definition of the PageHistory constructor
                The PageHistory object stores a mru/lru (most and least recently used) list of size
maxSize pages for which you submitted a form
                When you add a page and the list is full we discard lru and add it as mru
                Pages are characterized by a guid and a a flag
                The flag indicates if the page has already been submitted
                This object has some methods that lets you add a page, update a flag for a page
and get a flag from a page.

================================================================== */
//no need to use prototypes since we use only one instance of the object
function PageHistory(maxsize) {
        //define the member variables
        this.maxsize = maxsize;
        this.size = 0;
        //we store a js expression defining an associative array in the cookie
        //we instantiate the array for quick lookups in the js object pages
        //we keep the cookie to manage the lru/mru for serialization
        //this makes the whole thing much faster: remove beginning and add to the end are very easy ops
on strings
        //and we don't have to loop to serialize/deserialize
        this.cookie = getCookie("pageh");
        debug(GenDebug, "this.cookie", this.cookie);
        if (this.cookie == "") {
                this.pages = new Object();
        }
        else {
                this.pages = eval("bozo = " + this.cookie);
                this.size = this.cookie.split(',').length;
                debug(GenDebug, "this.size", this.size);
        }
        //define the methods
        this.setPageFlag = PageHistory_setPageFlag;
        this.getPageFlag = PageHistory_getPageFlag;
        this.isNotInHistory = PageHistory_isNotInHistory;
        debug(GenDebug, "pages", dumpObject(this.pages));
}

var pageHistory = new PageHistory(20);
```

FIG. 4

```
/*==========================================================================
        allowSubmit(name)

        Description:
            checks if a page is allowed to be submitted. If it is allowed to do so, returns true and sets its
flag to 0 so that it won't be allowed again.
                        to be called on a onSubmit handler

========================================================================== */
function allowSubmit(name, confirmFlag) {
   uKey = pageHistory.getPageFlag(name)
        debug(GenDebug, "uKey", uKey);
   if (uKey == "1") {
                pageHistory.setPageFlag(name, "0");
                return true;
   } else {
        if (confirmFlag) {
           return confirm("This form has previously been submited. Submitting again may result in an
error. Do you want to submit?");
        } else {
           alert("This Form has previously been submited. It cannot be submited again.");
           return false;
        }
   }
}

/*==========================================================================
        putPageInHistory(name)

        Description:
            puts a page in history if it is not there, with a flag allowing it to be submitted.
                        To be called at the beginning of your page

========================================================================== */
function putPageInHistory(name) {
        debug(GenDebug, "pageHistory.isNotInHistory(name)", pageHistory.isNotInHistory(name));
        if (pageHistory.isNotInHistory(name)) {
                        pageHistory.setPageFlag(name, "1");
        }
}

function dumpObject(obj) {
        var dump = "";
        for (var i in obj) dump += i + "=" + obj[i] + "\n";
        return dump;
}
```

FIG. 4

```
/*==================================================================
        getObject()

        Description: convert object name string or object reference into a valid object reference
        for both browsers: this is a reference on which you can set some style attributes
================================================================== */
function getObject(obj) {
        var theObj;
        if (typeof obj == "string") {
                var iniObj;
                if (isNav6) {
                        iniObj = document.getElementById(obj);
                }
                else {
                        iniObj = eval("document." + coll + obj);
                }

                if (IsUndef(iniObj)) {
                        return "undefined";
                }

                if (isNav4) {
                        return iniObj;
                }
                else {
                // in the IE or NS6 case the iniObj.style object may be undefined
                        if (IsDef(iniObj.style)) {
                                return iniObj.style;
                        }
                        else {
                                return "undefined";
                        }
                }
        }
        else {
                theObj = obj;
        }
        return theObj;
}
```

FIG. 4

```
/*==========================================================================
        getObjectRef()

        Description: convert object name string or object reference into a valid object reference
        for both browsers, without the style in IE: this is the real object reference
        this function is adapted from Danny Goodman's "Dynamic Html : The Definitive Reference"
        http://www.amazon.com/exec/obidos/ASIN/1565924940/qid%3D963012863/002-0174003-
8509633

========================================================================== */
function getObjectRef(obj) {//alert("getRef "+obj);

        var theObj;
        if (typeof obj == "string") {
                var iniObj = eval("document." + coll + obj);
//alert("getRef "+iniObj);
                if (IsUndef(iniObj)) {
                        return "undefined1";
                }
                        return iniObj;

                }
                else {
                theObj = obj;
                }
        return theObj;
}
```

FIG. 4

```
/* ====================================================================
FUNCTION:     IsUndef

INPUT:        val - the value to be tested

RETURN:       true, if the value is undefined
              false, otherwise.

PLATFORMS:    Netscape Navigator 3.01 and higher,
                  Microsoft Internet Explorer 3.02 and higher,
                  Netscape Enterprise Server 3.0,
                  Microsoft IIS/ASP 3.0.
==================================================================== */
function IsUndef( val ) {
    var isValid = false;
    if (val+"" == "undefined")
        isValid = true;

    return isValid;
} // end IsUndef

function IsDef( val ) {
    return !IsUndef(val);
} // end IsUndef

/**
* <pre>
* This function
*       checks if the form is allowed to be submitted.
*       if yes, it sets the action, then calls form.submit()
*
* Usage:
*
*  <code>
*       <a href="Javascript:submitForm('/iMM/somefile.jsp', 'myForm', 'someUniqueKey', true)">
*  </code>
*
* </pre>
*/
function submitForm(action, formName, key, confirmFlag) {
   //we put a provision here to let the users of this function not provide the last argument, which the
ndefaults to true
   if (IsUndef(confirmFlag)) {
        confirmFlag = true;
   }
   if (allowSubmit(key, confirmFlag)) {
        debug(GenDebug, "formName", formName);
        eval("document." + formName + ".action=\"" + action + "\"");
        eval("document." + formName + ".submit()");
   }
}
```

FIG. 4